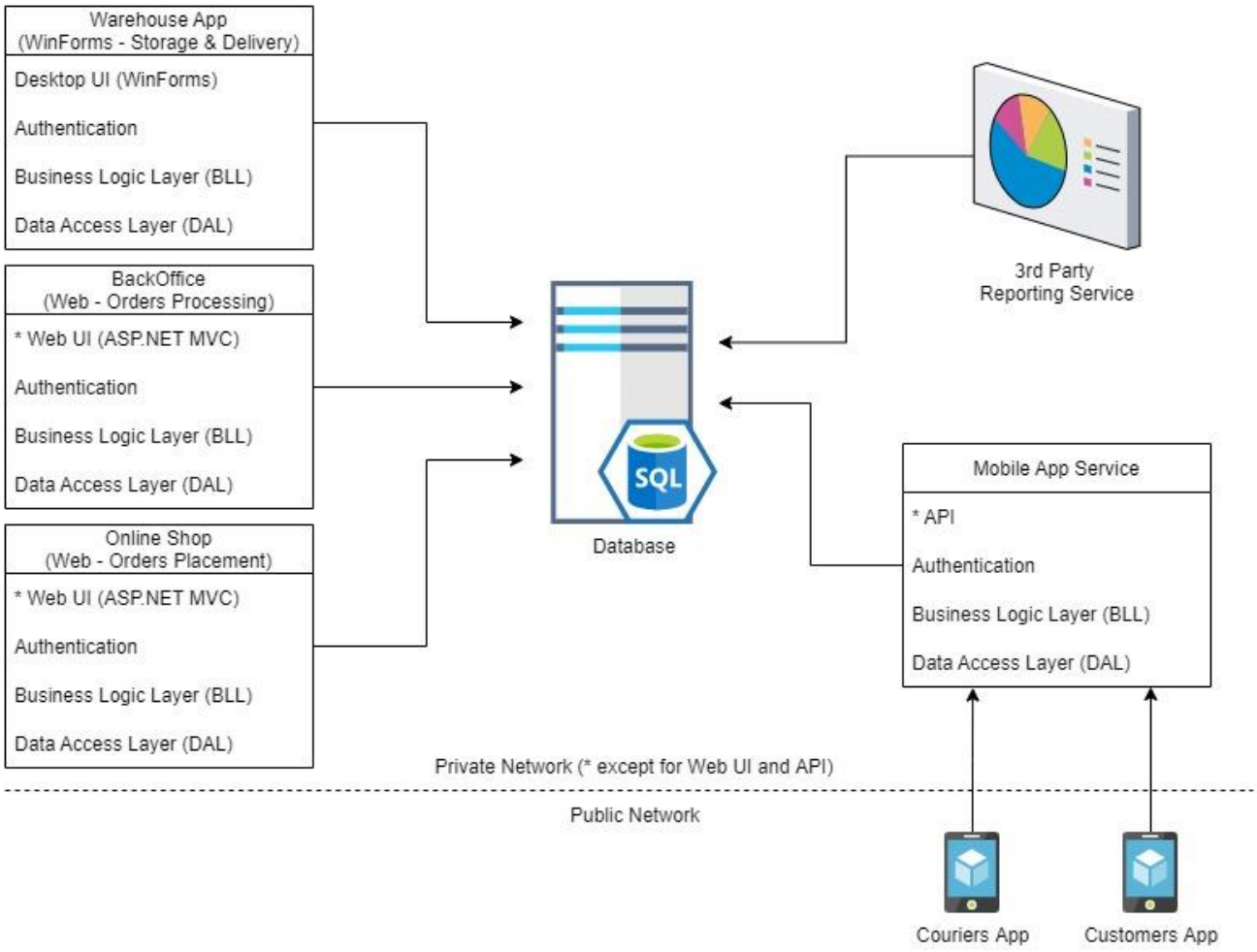
The customer is an online store with a warehouse and a delivery service. The architectural style is shown in Figure 1.

Figure 1 – Current architecture



The current architectural style is a set of monolithic applications that has a number of disadvantages and problem areas that are inherent in monolithic systems and that can arise as a project grows:

- difficult continuous deployment;
- difficult to scale;
- slow start-up time;
- etc.

Customer has plans to enter the markets in several other countries. This means that the client base will grow, and with it the need for features will grow too. It is also necessary to maintain the current functionality in working order. Also, with the entry into new markets, it will become necessary to deploy a new infrastructure, which will be problematic and costly with the current architecture.

To select a new architectural style, more input data is needed, such as:

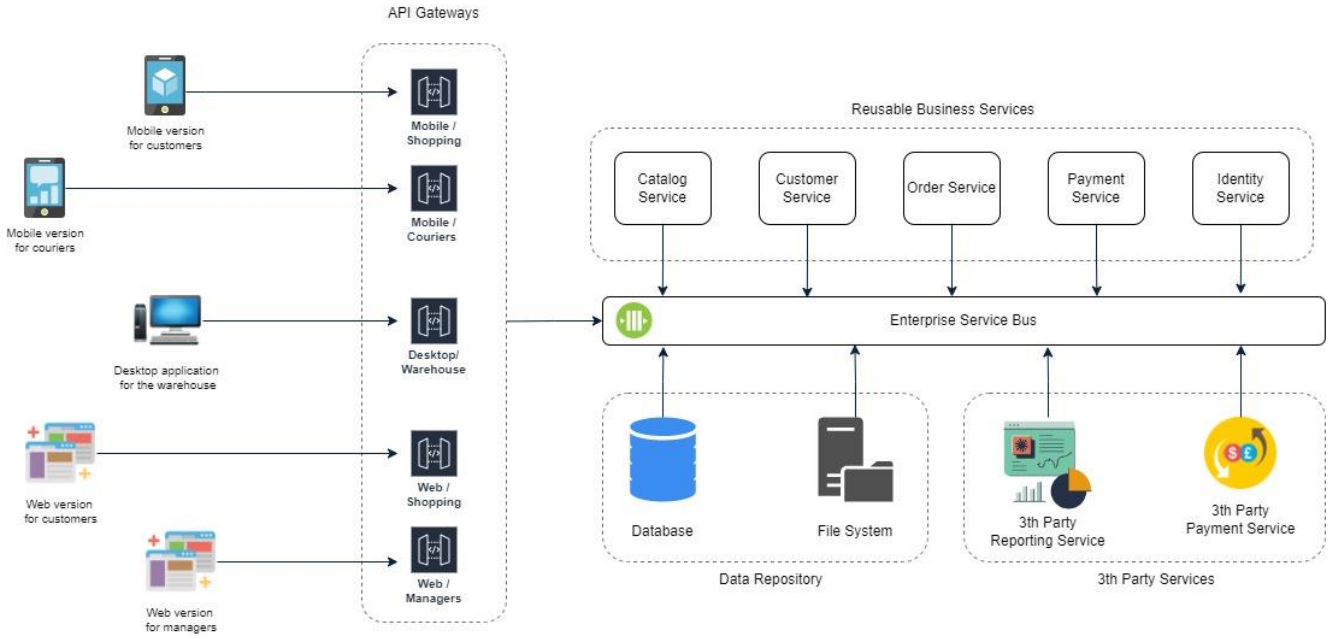
What are the sales forecasts in the new country? From this question, we should understand whether we want to invest large resources (developers, testers, cloud rent) from the start or not.

It is logical to assume that we would like to get a better result with less investment. Ultimately, we would like to have an architecture with the following advantages:

- Easy to scale;
- Testable;
- Ease of understanding = simple support;
- Smaller and faster deployments.

To meet all these requirements, I would choose a service-oriented style of architecture, but with the possibility of switching to a hybrid style. We can switch to a hybrid style gradually, depending on the new requirements and depending on project growth. The proposed architectural style is shown in Figure 2.

Figure 2 – Current architecture



API Gateways are used as an aggregation layer. The presence of such a layer helps to abstract from internal implementation and simplify the transition to a microservice architecture if necessary.

The main advantages of this architecture:

- Reusable services. Helps to avoid copy/paste, making future support easier and cheaper;
- Possibility of parallel development. A separate team can develop a separate service;
- Simplified CI/CD pipeline. Each service can be delivered separately;
- Scalable. If any service getting many users, then it can be easily scalable by attaching more servers.

The main advantages of this architecture:

- The ESB could become a single point of failure which impacts the entire system. Since every service is communicating through the ESB, if one of the services slows down, it could clog up the ESB with requests for that service.